WHAT IS CLAIMED IS:

1. A method comprising:

requesting access to a shared resource for a first process having a first local priority;

determining if a second process is simultaneously requesting access to the shared resource, the second process having a second local priority; and

if the second process is simultaneously requesting access to the shared resource, then granting access one of the first priority and the second priority having a higher local priority.

2. The method of claim 1, wherein the local priority is fixed for each of the first and the second process.

3. The method of claim 1, additionally comprising if the second process is not simultaneously requesting access to the shared resource, then:

determining if the second process currently has a lock on the shared resource;

if the second process currently has a lock on the shared resource, then denying the first process access to the shared resource; and

if the second process does not have a lock on the shared resource, then granting the first process access to the shared resource.

4. A method comprising:

requesting access to a shared resource for a first process having a first local priority, and a first wait time;

determining if a second process is simultaneously requesting access to the shared resource, the second process having a second local priority, and a second wait time;

if the second process is simultaneously requesting access to the shared resource, then granting access to one of the first process and the second process having a longer wait time; and

if the first wait time equals the second wait time, then granting access to one of the first process and the second process having a local higher priority.

5. The method of claim 4, wherein the local priority is fixed for each of the first and the second process.

6. The method of claim 4, additionally comprising if the second process is not simultaneously requesting access to the shared resource, then:

determining if the second process currently has a lock on the shared resource;

if the second process currently has a lock on the shared resource, then denying the first process access to the shared resource; and

if the second process does not have a lock on the shared resource, then granting the first process access to the shared resource.

7. A method comprising:

requesting access to a shared resource for a first process having a first global priority on a global priority queue of a global arbiter;

determining if a second process is simultaneously requesting access to the shared resource, the second process having a second global priority on the global priority queue of the global arbiter; and

if the second process is simultaneously requesting access to the shared resource, then granting access to one of the first process and the second process having a higher global priority.

8. The method of claim 7, wherein the global priority queue is one of a plurality of global priority queues in the global arbiter, and each global priority queue corresponds to a given shared resource.

9. The method of claim 7, additionally comprising if the second process is not simultaneously requesting access to the shared resource, then:

determining if the second process currently has a lock on the shared resource;

if the second process currently has a lock on the shared resource, then denying the first process access to the shared resource; and

if the second process does not have a lock on the shared resource, then granting the first process access to the shared resource.

10. A method comprising:

requesting access to a shared resource for a first process having a first global priority on a global priority queue of a global arbiter, and having a first wait time;

determining if a second process is simultaneously requesting access to the shared resource, the second process having a second global priority on the global priority queue of the global arbiter, and having a second wait time;

if the second process is simultaneously requesting access to the shared resource, then granting access to one of the first process and the second process having a longer wait time; and

if the first wait time is equal to the second wait time, then granting access to one of the first process and the second process having a higher than global priority.

11. The method of claim 10, wherein the global priority queue is one of a plurality of global priority queues in the global arbiter, and each global priority queue corresponds to a given shared resource.

12. The method of claim 10, additionally comprising if the second process is not simultaneously requesting access to the shared resource, then:

determining if the second process currently has a lock on the shared resource;

if the second process currently has a lock on the shared resource, then denying the first process access to the shared resource; and

if the second process does not have a lock on the shared resource, then granting the first process access to the shared resource.

13. An apparatus comprising:

a local arbiter to arbitrate on behalf of the corresponding process for one of a plurality of resources; and

a semaphore to indicate a status of the corresponding process.

14. The apparatus of claim 13, additionally comprising a local priority block to indicate a local priority of the corresponding process.

15. The apparatus of claim 13, additionally comprising a timer element to determine a wait time for the corresponding process.

16. A system comprising:

one or more shared resources; and

one or more processes, each corresponding to a semaphore system, and each semaphore system having a local arbiter to arbitrate for access to a given one of the shared resources.

17. The system as in claim 16, wherein a given semaphore system additionally comprises a local arbiter block having a local priority corresponding to a corresponding process, and the local arbiter arbitrates for access to a given one of the shared resources by granting access to the corresponding process if its corresponding process has a local global priority than a conflicting process.

18. The system of claim 17, wherein the local priority is fixed.

19. The system as in claim 17, wherein the given semaphore system additionally comprises a timer element, and the local arbiter arbitrates for access to a given one of the shared resource by:

granting access to the corresponding process if the corresponding process waited longer for the given resource than the conflicting process; and

if the corresponding process waited the same amount of time for the given resource as the conflicting process, then granting access to the corresponding process if the corresponding process has a higher local priority than the conflicting process.

20. The system of claim 16, wherein the local arbiter arbitrates for access to a given one of the shared resources by granting access to the corresponding process if there are no conflicting processes.

21. The system as in claim 20, the system additionally comprising a global arbiter having a global priority queue, the global arbiter to:

modify process priorities by moving processes that have been granted access to a given resource to a position in the global priority queue having a lowest priority; and

arbitrate conflicts between a first process and a second process by granting access to one of the first process and the second process having a having a higher global priority.

22. The system of claim 16, wherein the semaphore additionally comprises a timer element, and the local arbiter arbitrates for access to a given one of the shared resources by:

granting access to the corresponding process if the corresponding process waited longer for the given resource than a conflicting process; and

if the corresponding process has waited the same amount of time for the given resource as the conflicting process, then offloading the arbitration process to a global arbiter.

23. The system as in claim 22, the system additionally comprising the global arbiter having a global priority queue, the global arbiter to:

modify priorities to processes by moving processes that have been granted access to a given resource to a position in the global

priority queue having a lowest priority; and

arbitrate conflicts between a first process and a second process by granting access to one of the first process and the second process having a higher priority.

24. A machine-readable medium having stored thereon data representing sequences of instructions, the sequences of instructions which, when executed by a processor, cause the processor to perform the following:

request access to a shared resource for a first process having a first local priority;

determine if a second process is simultaneously requesting access to the shared resource, the second process having a second local priority; and

if the second process simultaneously requests access to the shared resource, then grant access one of the first priority and the second priority having a higher local priority.

25. The machine-readable medium of claim 24, wherein the local priority is fixed for each of the first and the second process.

26. The machine-readable medium of claim 24, additionally comprising if the second process is not simultaneously requesting access to the shared resource, then additionally comprising sequences of instructions which, when executed by a processor, cause the processor to perform:

determine if the second process currently has a lock on the shared resource;

if the second process currently has a lock on the shared resource, then deny the first process access to the shared resource; and

if the second process does not have a lock on the shared resource, then grant the first process access to the shared resource.

27. An apparatus comprising:

at least one processor; and

a machine-readable medium having instructions encoded thereon, which when executed by the processor, are capable of directing the processor to:

request access to a shared resource for a first process having a first local priority;

determine if a second process is simultaneously requesting access to the shared resource, the second process having a second local priority; and

if the second process simultaneously requests access to the shared resource, then grant access one of the first priority and the second priority having a higher local priority.

28. The apparatus of claim 27, wherein the local priority is fixed for each of the first and the second process.

29. The apparatus of claim 27, additionally comprising if the second process is not simultaneously requesting access to the shared resource, then additionally encoded instructions which, when executed by a processor, are capable of causing the processor to:

determine if the second process currently has a lock on the shared resource;

if the second process currently has a lock on the shared resource, then deny the first process access to the shared resource; and

if the second process does not have a lock on the shared resource, then grant the first process access to the shared resource.

30. An apparatus comprising:

means for arbitrating on behalf of the corresponding process for one of a plurality of resources; and

means for indicating a status of the corresponding process.

31. The apparatus of claim 30, additionally comprising means for indicating a local priority of the corresponding process.

32. The apparatus of claim 30, additionally comprising means for determining a wait time for the corresponding process.